HIGH-YIELD DECISION RULES

CCA-F Foundations - Quick Review Sheet

Use each card as a single-best-answer filter: identify the governing rule, then reject the common trap.

Agentic loop

Rule: Continue on `stop_reason: tool_use`; execute requested tools, append `tool_result`, and terminate normally on `end_turn`.

Trap: Assistant prose or a fixed cap is not the primary stopping signal.

Deterministic enforcement

Rule: Use hooks or programmatic gates for mandatory prerequisites and policy limits.

Trap: Prompt wording and examples reduce risk but do not guarantee compliance.

Coordinator pattern

Rule: Use hub-and-spoke orchestration for delegation, routing, aggregation, recovery, and observability.

Trap: Peer-to-peer agents create hidden state and inconsistent error handling.

Subagent context

Rule: Pass facts, sources, constraints, and output criteria explicitly; scope tools to the role.

Trap: Subagents do not inherit parent history or share memory automatically.

MCP interfaces

Rule: Use resources for catalogs and reference content; tools for actions. Give tools distinct names and boundaries.

Trap: Overlapping descriptions and too many tools degrade selection.

MCP errors

Rule: Differentiate transient, validation, permission, and business failures; distinguish valid empty results from access failures.

Trap: Generic errors, silent empty success, and blind retries hide the recovery path.

Claude Code scope

Rule: Use project scope for team standards, user scope for personal preferences, and path-scoped rules for conditional conventions.

Trap: A user-level file is not shared through version control.

Plan vs direct

Rule: Use plan mode for multi-file, architectural, or multi-approach work; direct execution for small, clear changes.

Trap: Do not start implementation first when complexity is already known.

Prompt precision

Rule: Define explicit reportable and excluded categories; use few-shot examples for ambiguous boundaries.

Trap: 'Be conservative' and generic confidence thresholds do not define correctness.

Structured output

Rule: For the exam, use schema-backed `tool_use` and the appropriate `tool_choice`; validate semantics separately.

Trap: Schema validity guarantees shape, not factual or business correctness.

Batch strategy

Rule: Use Message Batches for non-blocking, latency-tolerant work and `custom_id` for correlation; keep blocking flows synchronous.

Trap: Cost savings do not create a latency guarantee.

Context

Rule: Persist exact facts, trim verbose tool outputs, surface summaries early, and isolate exploration with subagents or scratchpads.

Trap: Do not lose amounts, dates, IDs, or source metadata in vague summaries.

Escalation and review

Rule: Honor explicit human requests, escalate policy gaps or inability to progress, and calibrate field-level review using labeled data and stratified sampling.

Trap: Sentiment and generic self-confidence are weak proxies for case risk.

Provenance

Rule: Preserve claim-source mappings, dates, methodology, conflicts, and coverage gaps through synthesis.

Trap: Do not average, suppress, or arbitrarily choose between credible conflicting claims.

VERSION-AWARE PREPARATION

Exam Baseline vs Current Documentation

For exam preparation: follow the terminology and task framing in the stated exam-guide baseline. For production work, verify the latest official documentation before implementation.

Area	Exam-guide framing used in this pack	Current-docs caution
Item format	The public Version 1.0 guide, effective July 2026, lists multiple-choice and multiple-response items.	This pack intentionally remains single-best-answer and therefore does not replicate every current item type.
Structured output	`tool_use` plus JSON schemas, with `tool_choice` used to guarantee a tool call when needed.	Current Claude documentation also describes dedicated Structured Outputs features. Treat them as production updates, not replacements for the guide framing.
Subagent invocation	The guide names the `Task` tool and requires the coordinator to have permission to invoke it.	Current Claude Agent SDK documentation uses the `Agent` tool and notes that it was renamed from `Task`.
MCP errors	Use `isError` plus structured categories, retryability, readable detail, and partial results where useful.	The MCP specification distinguishes protocol errors from tool execution errors; fields such as `isRetryable` remain application-level conventions.

Scenario Decision Lens

01 Customer Support
Agentic loops, deterministic policy controls, tool selection, ambiguity, and escalation.

02 Claude Code
Configuration scope, rules, skills, plan mode, iterative refinement, and session handling.

03 Multi-Agent Research
Coordinator-subagent orchestration, explicit context transfer, errors, provenance, and conflicts.

04 Developer Productivity
Built-in tool choice, codebase exploration, scoped MCP access, state persistence, and safe edits.

05 CI/CD
Headless CI, structured findings, precision, independent review, multi-pass analysis, and batching.

06 Structured Extraction
Schema design, few-shot extraction, validation, context, confidence calibration, and provenance.

Version discipline. This pack is aligned to Claude Certified Architect - Foundations Exam Guide, Version 1.0, effective July 2026 and versioned independently as 1.0.1. Official links are maintained in `SOURCES.md`.